Data Structures Assignment #1

The goal of this assignment was to create a data structure that models a mathematical set of integers. The goal of this data structure is to be able to:

* Insert new items quickly
* Delete items quickly
* See if an item is in the set quickly
* See if a given set is a subset
* Generate the union of 2 sets
* Generate the intersection of 2 sets

The next factor to consider is that this is implemented in C++. In general, the reason to program in C++ is to create fast software. With these considerations in mind, I wrote my set implementation with most operations having O (log n) execution time.

This was achieved by writing my own dynamic array. It grows by twice the size each time it is recreated. Therefore, memory only needs to be reallocated O (log n) times.

The source code is meant to be compiled as a *project*. A compiled binary for windows has been included for convenience.

The set does not use templates; however, it uses a 'typedef int set\_type'. This means it could easily be changed to work with any data type that implements the '<' operator.

The set applies the Rule of 3 <http://en.wikipedia.org/wiki/Rule_of_three_%28C%2B%2B_programming%29>

In addition to the destructor, copy constructor, and copy assignment operator, the equality operators are also implemented.

I stress tested the set against std::set<int> and my set inserted and looked up 10,000,000 elements nearly 4.5 times faster. In addition, for 100 million elements, my set succeeded while the std::set ran out of memory. I suspect the performance difference comes from the fact that the Standard Template Library makes heavy use of the iterator design pattern and returns an iterator for each operation.